Intel Architecture Labs

Intel® Real-Time Performance Analyzer User Guide

Version 1.0

September 22, 2000

1. Introduction

The Intel® Real-Time Performance Analyzer (hereafter referred to as simply 'the performance analyzer') enables you to analyze and characterize the kernel-level performance and behavior of a computer system using the Windows* operating system. You can either test the statistical behavior of the system for a long period of time (24 hours or so), or examine the exact behavior of the system for a short period of time (up to about 20 minutes).

1.1 System Requirements

Recording data requires an IA-32 Intel-based single processor desktop system running on an Intel® Pentium™ Pro processor or later. This includes the Intel® Pentium™ II, Pentium® III, Pentium® 4, or Celeron™ processor. You can view saved data on any computer capable of running the Microsoft* Windows NT* 4.0 (or later) operating system. The Intel® Real-Time Performance Analyzer tool is compatible with Windows* 98 Second Edition, Windows Millennium Edition, Windows NT 4.0 (with at least service pack six installed), or Windows 2000 operating systems. The system must contain at least 128 MB of system memory. A minimum of 4 GB of free disk space is recommended. For best results use a separate disk for temporary files.

1.2 Errata

The following are known errata with the Intel® Real-Time Performance Analyzer:

- It is unable to run at the same time as Intel® VTune™ performance tool release 4.0 or earlier.
- It cannot be executed at the same time as USB audio with Windows Millenium as the operating system.
- Multiple processor systems, notebook / laptop systems are not supported
- It cannot coexist with any other software that uses the local APIC (Advanced Programmable Interrupt Controller) or Pentium processor performance counters.

2. Installation

To install the Intel® Real-Time Performance Analyzer, run SETUP.EXE from the web download.

2.1 Installation Options

If you are installing the Intel® Real-Time Performance Analyzer on a system that boots two or more versions of the Windows operating system, the two installations must be based in different Program Files directories. Otherwise, uninstalling one will also uninstall the other.

After accepting the license agreement and installation location, you will be prompted for the type of installation. There are three choices:

- **Custom:** This installation allows you to choose which elements to install. There are four elements to the program:
  - **Program Files:** The two main elements to logging and analyzing the system are the driver and the viewer. The driver interfaces with the Windows kernel and analyzes it. The viewer displays this data on the screen. This option installs the viewer. This program can be used when the driver is not installed, but only to look at data files that have been generated on a system that has the driver installed.
  - **Driver Files:** This installs the driver. If this option is selected, the Program Files option must also be selected. The install program enforces this restriction.
  - **Help files:** This option installs the user guide for the performance analyzer.
  - **Load simulator:** This option installs a program that allows artificial tasks to be created. It is only available on Windows 2000 and Windows NT 4.0.
- **Viewer only:** This option installs the Program Files and the Help Files.
- **Full installation:** This option installs all four options

After installation is complete, restart the PC system to use the program.

2.2 Uninstallation

To uninstall the Intel® Real-Time Performance Analyzer select Add/Remove Programs from the Windows Control panel and then select the performance analyzer's entry. You will be given the opportunity to repair or modify an existing installation or to remove the program entirely.

3. Concepts and Usage

The Intel® Real-Time Performance Analyzer creates and displays log files of the system's activities. It can also create simulated workloads to measure the desktop PC system under various loads. Here are a few basic patterns of use:

- You know that a certain activity causes the system to lock up for so long that your device driver or hardware is unable to get the processor time that it needs. You want to find out what is causing this, and/or for exactly how long the system is unavailable.
- You want to examine the system to find out what the latencies for various kernel events are like under various loads.
- Someone else has already done this work and you want to look at the results.

3.1 System Analysis

3.1.1 Recording Data

One of the performance analyzer's most powerful features is the ability to examine the processor scheduling activities in the system at a very fine level of detail. For each event that the performance analyzer can measure, it gathers a data sample that contains information about that event. Here are the items in which the performance analyzer can measure. Unless noted otherwise, all measurements are accurate to the clock cycle:

- **Interrupt Service Routines (ISRs):** When an external interrupt comes in, the performance analyzer can capture and display when the interrupt first came in and when the ISR finished executing. This functionality can only see interrupts that were hooked after the driver was started. The default installation installs it as a boot driver, which is the earliest possible time for a driver to start.
- **Deferred Procedure Calls (DPCs):** When a timer DPC is queued, the performance analyzer can capture and display when it starts and finishes executing. When a non-timer DPC is queued, the performance analyzer can see when it is queued, when it starts executing, and when it is finished executing. The performance analyzer can also report the address from which the DPC is queued, and the importance of the DPC.
- **Work Items:** When a work item is queued, the performance analyzer can capture and display when the work item is inserted, started, and finished. It can also report the address from which it is queued and the importance of the work item.
- **Clearing and setting the interrupt flag (CLI/STIs):** Whenever a process does a CLI/STI pair, the performance analyzer can log this. These events are logged as Interrupt Delays. The performance analyzer does this measurement by causing the performance counter to trigger when 1024 clock cycles have elapsed with the interrupt flag cleared. Because of this, the start address and time can be off by up to 1024 clock

cycles, and a very short CLI/STI pair can be missed entirely. End times and addresses are correct.

- **HIGH_IRQL periods:** Whenever the IRQL level is raised to High (31), the start and stop times are logged. The address of the code that raised the IRQL is also logged.

To log any of these five items, start the performance analyzer. When the initial dialog comes up, select Create a new session. You will be presented with a dialog that offers all of the system logging options. Select the various checkboxes to indicate what you want to record.

When an item is selected, the default behavior is to record all such items that occur. You can reduce the amount of data recorded by only recording items for which the timing exceeds a certain threshold. Figure 1 shows an example of recording all DPCs, all high-IRQL periods, and all ISRs that execute for more than 5 µs.

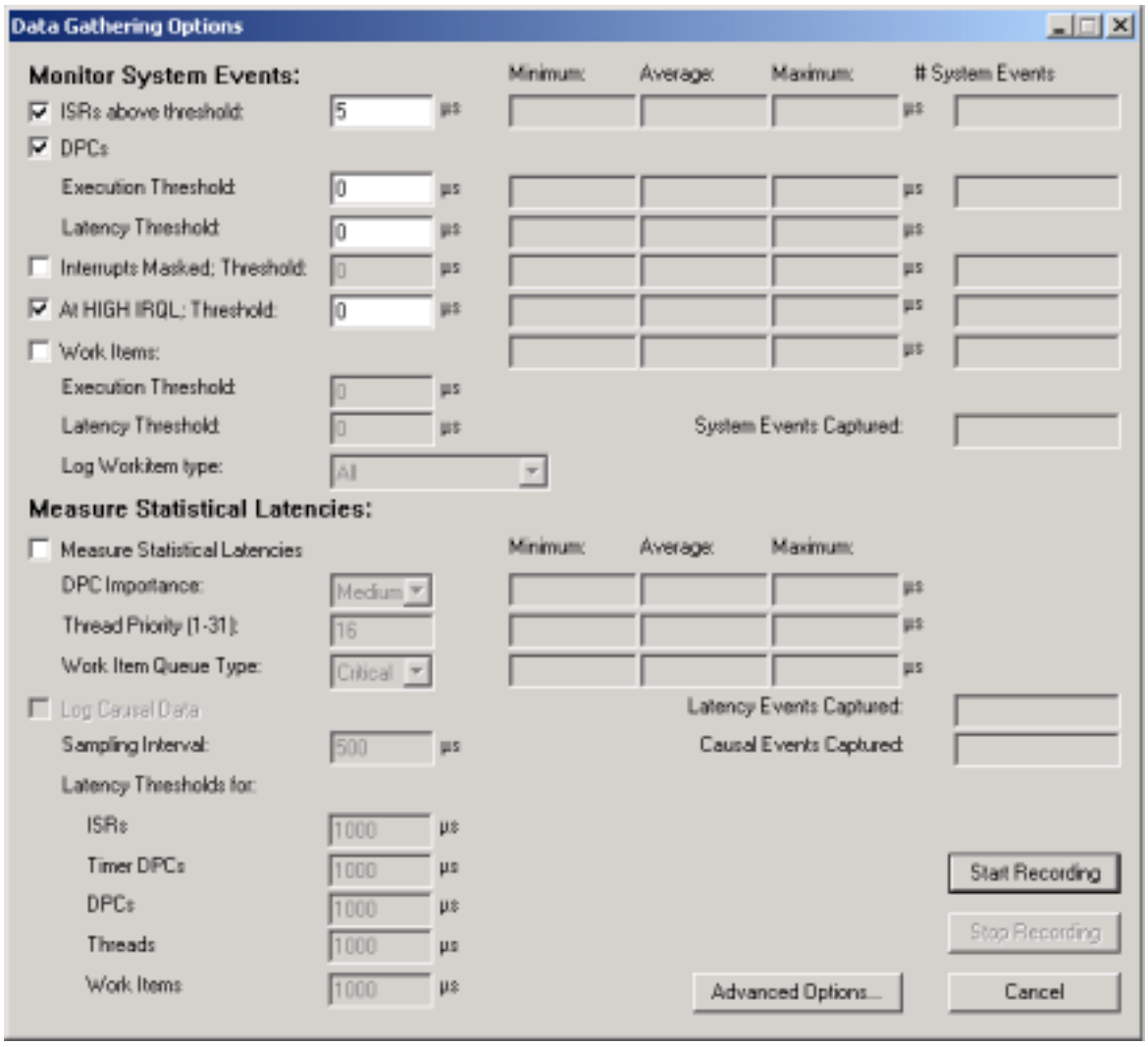

Figure 1. Example of the Recording Dialog

After selecting the items to record, click on Start Recording to begin the recording activity. The dialog will show you statistics on what is being recorded. These statistics include the minimum, average, and maximum of each data point. After gathering the required data, click Stop Recording. Each sample is 96 bytes, and there is a 2 billion byte limit of gathered samples. This equates to a little over 20 million samples.

3.1.2 Viewing Recorded Data

When done recording, a list view appear showing the samples that were gathered, as shown in Figure 2. By default, this view shows the first 2000 samples. This shows a general view of every sample, including type and length information. To view the details of a specific type of sample, use the selection box at the top of the window to select which kind of sample to display. For example, when you select DPC, all DPCs within the first 2000 samples gathered are displayed showing much more information than is shown in the general view.

You have the option to display a sample in the timeline view (see Section 3.1.3). By right clicking on the sample, a popup menu will appear. The only option on it is View in Timeline. Upon selection of this option, a timeline appears, centered around the selected sample. To view other samples, enter a number in the Get Sample box at the top of the window, and press the Enter key or click Get Sample. This causes the current list of 2000 samples to be centered on the sample selected. You can center the view on any sample that is currently displayed by double-clicking it in the main window. The length of the loaded list can be changed in the Tools→ Program Options dialog (see Section 4.2).

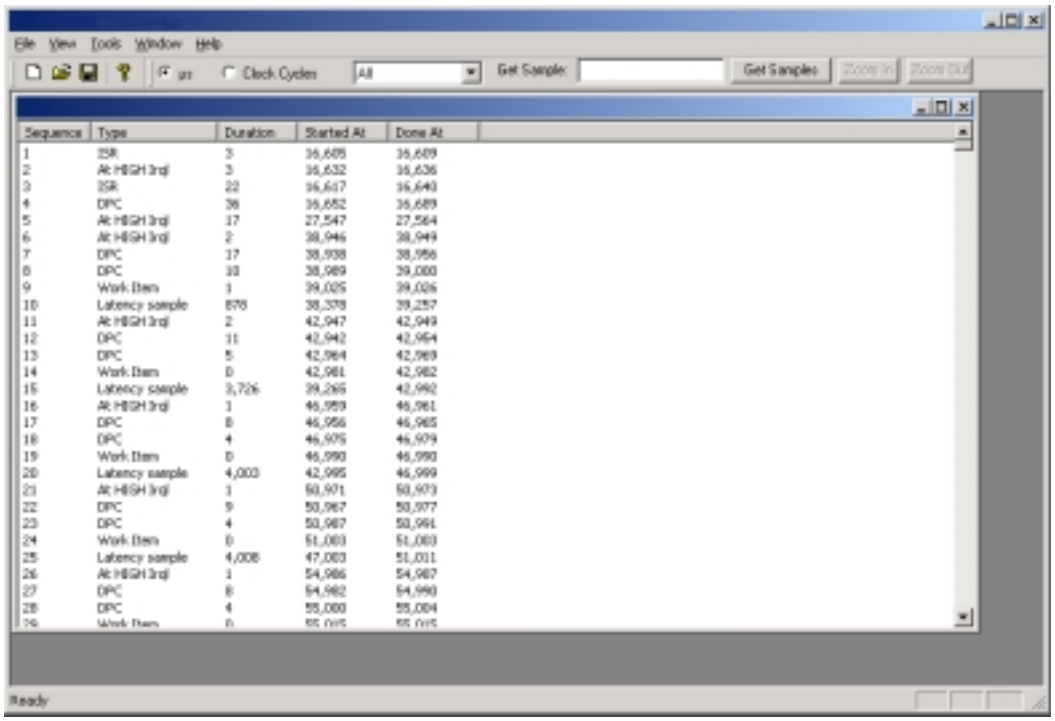

Figure 2. The List View

While this view is useful, it is difficult to get a picture of the overall data. There are many other views accessible from the View menu that can provide a better view of the complete data set.

3.1.3 Timeline View

A key feature of the performance analyzer is the ability to graphically view the system's activities. Once a set of samples is recorded, select View→Timeline View. This creates a timeline of the operating system's activity during the recording session (see Figure 3).

There are two different timeline views. The first is the macro view. This view divides the recording session (or a subset of it) up into blocks and displays the longest of each kind of sample during a specific time. The default view shows all samples, but the selection box at the top of the window can be used to filter only a certain kind of sample. If a driver marker appears in a particular block, a small red flag appears at the top of the graph.

There are two small arrows at the top of the graph. These start out at the upper left and right corners of the graph. You can relocate them by left- or right-clicking on the graph where you want the left and right side, respectively. If you click and drag on the graph with either button, a solid line will show where the selected point will be. This enables you to place the boundaries exactly where you want them. After relocating the arrows, click Zoom In for a closer look at the graph. The graph will be redrawn, with the left and right sides now corresponding to where the arrows were. This allows you to zoom in on a point of interest (an extra long DPC, for example) very quickly.

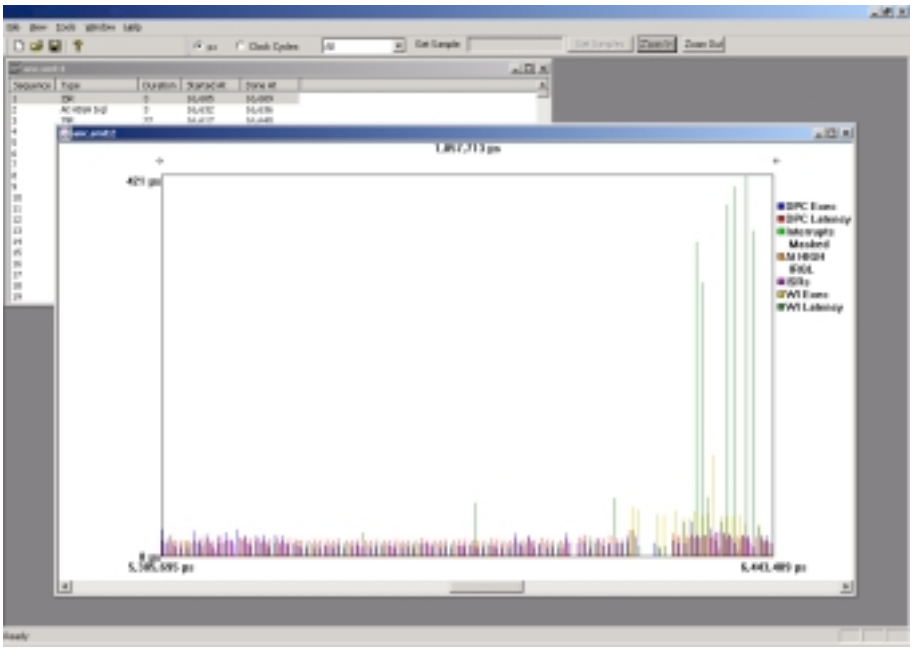

Figure 3. The Timeline View

Figure 3 shows the same document loaded in Figure 2, but with a timeline view added. This view is a little past the halfway point of the entire document, as can be seen by the scrollbar at the bottom. The single longest sample in the graph is 421 µs, and is a work item latency. The entire graph is approximately 1.06 seconds wide.

After zooming in close enough so each individual pixel is less than 100 µs wide, the view changes to the micro view. The default value (100 µs) can be changed through the Tools→Program Options menu item. This view is a timeline of the operating system's activity. Individual DPCs, work items, interrupts, interrupt masked periods, driver markers, and raised IRQLs are shown in this view. The height at which a particular sample is shown depends on its IRQL. DPCs run at IRQL 2, work items at IRQL 0, ISRs at variable levels and high IRQLs at 31. Because CLI/STIs (interrupt delays) can run at any level, they are shown across the bottom of the graph.

A sample consists of a thick horizontal line with thin vertical lines on the end to allow you to distinguish overlapping samples. If the sample is a work item or a non-timer DPC, a gray X is drawn at the queuing point, and then the line is drawn showing where the actual execution was. When the pointer is over the execution line, the queuing X turns red.

To learn more about a particular sample, place the pointer over the sample. A ToolTip will pop up displaying more information about the sample. When you double-click a sample a new list view is created, centered on the sample.

Figure 4 shows a zoomed in view of the same data file. The window is now only 161 µs wide. At the bottom of the window (1) is a Work Item that has been active since before the left side of the window and will continue to be active past the right side of the window. Work Items run at the lowest priority of any kernel-mode code and thus are preempted by all other items that the performance analyzer records. On this view, a Timer DPC (2) started approximately at the 20 µs mark. Either the Timer DPC or an unrecordable ISR that preempted it queued a DPC (3) about 5 µs into it. The DPC was preempted by an ISR (4) at IRQL 18, which ran for about 2 µs. A second ISR (5) came in about 4 µs later. This second ISR queued another DPC (6) and then for a short time raised the IRQL to 31 (7). After that ISR finished executing, the IRQL was raised again for a short period of time (8). Not long after that, the DPC that was queued at (6) began executing (9), and when the DPC was completed, the DPC that was queued back at (3) began executing (10). The DPC (10) executed after (9) because it was a lower priority even though it was queued first. In this example, (9) is associated with (6) because the X at (6) turns red when the pointer is over the bar at (9). When this data file was recorded, Interrupt Delay information was not gathered. Had it been gathered, they would have appeared across the bottom of the graph, just below the work item at (1).

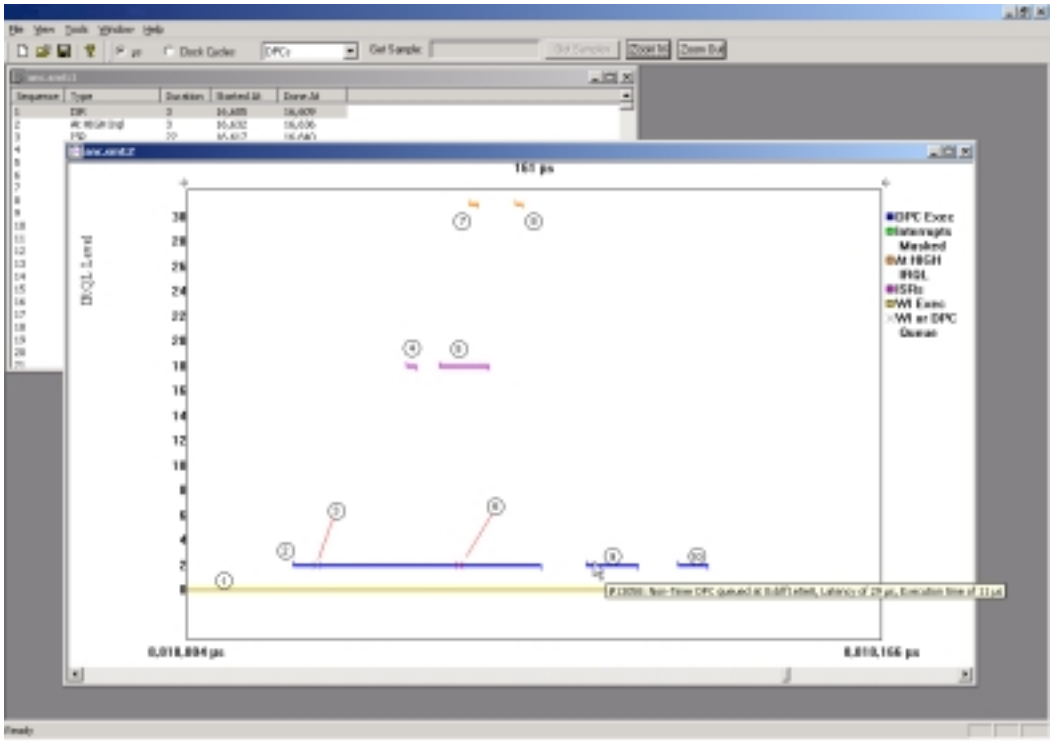

Figure 4. Close-up View of a Sample Data File

Finally, a note about Work Items: Work Items can be very long, lasting several seconds or more. In the macro view (bar graph) of the timeline view, the yellow bar representing a work item execution is drawn at the column corresponding to the end of its execution period. However, performance optimizations cause long work items to sometimes not be displayed unless you are zoomed in near the beginning of the sample. Because of this, you will sometimes need to zoom in reasonably close to a work item, and still in the macro view, make a note of where it starts and ends, and from there know that there is a work item executing during that time. Work Items can be pre-empted and are non-exclusive, so they have minimal effect on the other items that the performance records. In theory, other items, were they to stretch out that far, could also cause these problems. In practice, only Work Items take this long to execute.

3.1.4 Other Views

Figure 5 is an example of a bucketed view. This shows data similar to the charted view, but it is text-based and plotted in equally sized buckets. You can choose what size the buckets are and how many will be plotted. For example, if you choose 1000 1-millisecond buckets, there will actually be 1001 buckets created. The first for <1 ms, the next for <2 ms, and so on, up to the last one, which is for >1000ms. You can export this data to a text file by selecting Tools→Export Bucketed Data.

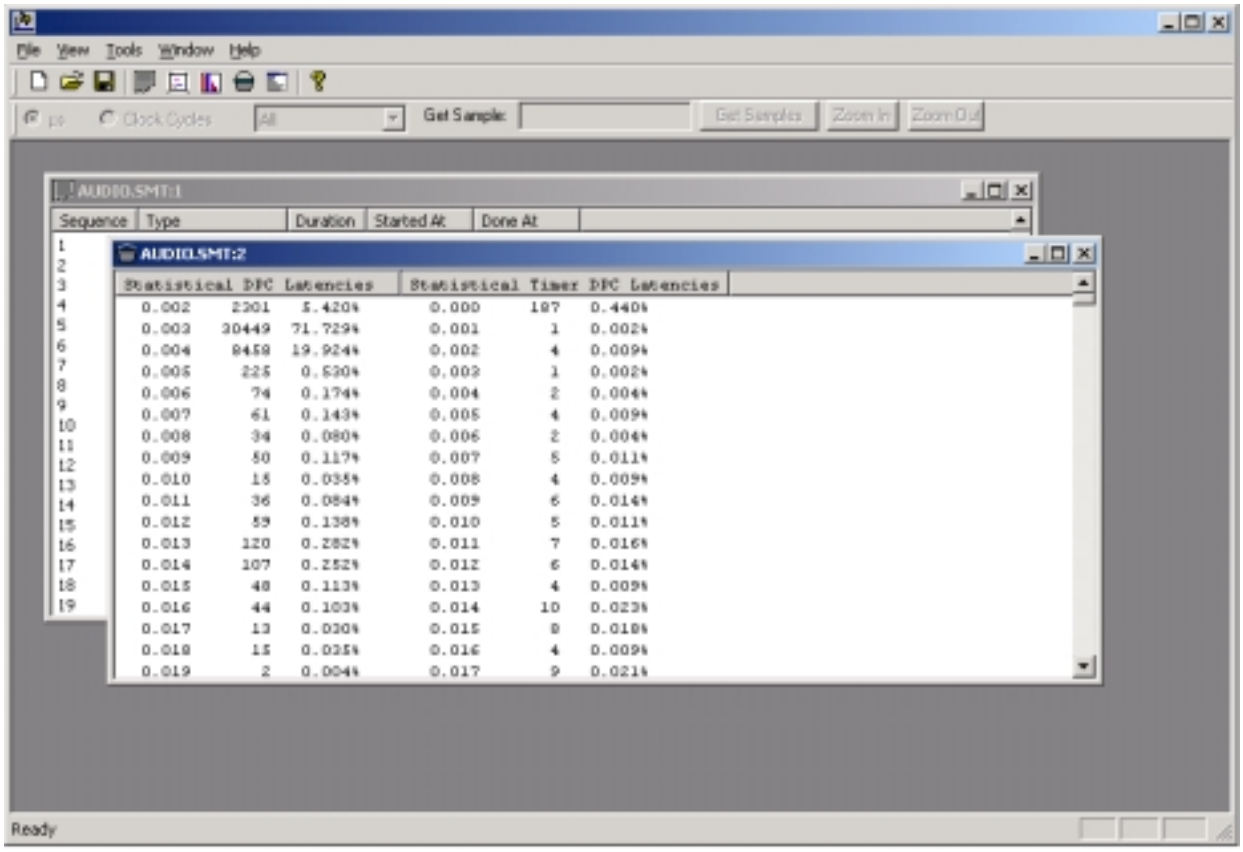

Figure 5. Example of the Bucketed View

3.2 Statistical Analysis

3.2.1 Overview

The statistical analysis feature of the Intel® Real-Time Performance Analyzer characterizes the worst-case latencies in different levels of the Windows operating system scheduling hierarchy. This information can be used to analyze the real time performance of the operating system under load and project the quality of service a soft device driver can expect from the system. The theory behind such a measurement is presented in *A Comparison of Windows Driver Model Latency Performance on Windows NT and Windows 98* available at: ftp://download.intel.com/ial/sm/osdi1999.pdf.

The latency measurements capture the ability of Windows to support multimedia and real-time workloads. The measurements abstract the services provided by the Windows Driver Model (WDM) into an operating system scheduling hierarchy as follows:

- Interrupt Service Routines (ISRs). Execute at IRQLs from dispatch to high level (31)
- WDM Deferred Procedure Calls (DPCs). FIFO/LIFO queue depending on DPC priority (high, medium, and low importance)
- Real-time Priority Threads. Timesliced, execute at Win32 priorities 16 through 31

Each level of the operating scheduling hierarchy is fully preemptible by the level(s) above it (that is, with higher numbers), although in practice complications occur on Windows 98 since the legacy Windows 95 schedulers continue to exist.

The operating system latencies measured include the following:

- Interrupt latency: Delay from the assertion of the hardware interrupt, as seen by the processor, until the first instruction of the software interrupt service routine (ISR) is executed. The performance analyzer measures the ISR latency of the PIT (timer) ISR. Since this is a high level interrupt, this measurement mostly reflects the latency introduced by interrupt holdoffs.
- Timer DPC latency: Delay from when the timer is supposed to queue a DPC until the first instruction of the DPC is executed. By default, the performance analyzer queues a timer DPC to execute after a delay of 4 ms. This measurement mostly captures the latency introduced by interrupt holdoffs in the delay interval and all ISRs executing in the interval between the timer expiration and the DPC execution.
- DPC latency: Delay from the time at which the Timer DPC queues a DPC until the first instruction of the DPC is executed. Since the system is already at dispatch level when this DPC is queued this measurement captures only the latencies introduced by DPCs ahead of this DPC in the DPC queue and any interrupts that occur in that short interval.
- WorkItem latency: Delay from the time at which the DPC queues a WorkItem until the time at which the WorkItem executes the first instruction. This measurement includes the latencies introduced by other DPCs, higher real time priority threads and any interrupts that occur in that interval.
- Thread Latency: Delay from the time at which the DPC signals a waiting thread until the time at which the signaled thread executes the first instruction after the wait. This measurement includes the latency introduced by higher priority scheduling events, and the time required to save and restore thread context.

Statistical latency measurement is a much less invasive procedure than the system analysis. The sampling interval is determined by the Statistical Timer DPC delay option under Advanced Options in the configuration dialog box (default 4 ms). The performance analyzer drives all the latency measurements by queuing them from the timer DPC. At 250 samples per second (the default), the performance analyzer can run for about 24 hours before hitting the 2 GB limit mentioned in Section 3.1.1.

3.2.2 Recording the Data

To run a statistical analysis, create a new document and select the checkbox to record statistical data. You can select the priority of the DPC (Windows 2000 only), the priority of the thread and the priority of the Work Item. To begin, click Start Recording. When you've gathered the data that you need, click Stop Recording. The first view to come up will be the list view. Navigate the list view according to the directions in Section 3.1.2.

3.2.3 Viewing a Chart of the Data

The Intel® Real-Time Performance Analyzer has the ability to display charts of the gathered statistical data. To view a chart, select View→Add Charted View. This invokes a dialog asking what data to chart. Select the items to view and click OK. The charts will then be generated and displayed. This can be a very lengthy process if there is a lot of data, up to about five minutes.

As shown in Figure 6, the charts are double logarithmic. A double-logarithmic graph is one in which both the X- and Y-axis are based on logarithmic rather than linear scales. The first bar on the X-axis shows what portion of the gathered samples were between zero and 124.999 µs in length. The next bar shows all samples between 125 µs and 249.999 µs, and the bar after that shows everything between 250 µs and 499.999 µs. After that, the high value of each bar is twice that of the bar before it.

The height of each bar is related to what portion of the samples fall within its range. The top of the chart is 100%, and the bottom of the chart is 0.0001% (one part per million). However, halfway up the chart is not 50%, as it would be on a normal chart. Instead, the halfway point is 0.1%, which is 1000 times as much as the bottom and 1/1000th as much as the top.

For example, in a session with a million samples, where 998999 were at 50µs, 1000 were at 200 µs, and one was at 300 µs, the first bar would go virtually to the top of the graph, the second bar would go half way up the graph, and the final bar would be one pixel high.

These charts can be exported. Go to Tools→Export Charts to bring up a dialog enabling you to save these charts as bitmaps. After you select a filename, the saving routine saves the charts as that filename plus a designator of the chart type. See Section 4.4 for details on what the output charts will be called.

You can also compare multiple sessions. To compare two sessions, load one into the charted view and then select Tools→Import Dataset. Select the saved dataset to import into the graph and the two charts will be plotted side-by-side, in different colors. This allows the comparison of two different operating systems or workloads. Up to five documents can be charted together and compared.

Figure 6 shows an example of comparing two data files using the charted view. To generate this view, first load the file called dmaidle.smt into the main view, and then select View→Add Charted View. After the red chart is drawn, select Tools→Import Data Set and open audio.smt. This imports audio.smt into the same chart. In this example, the dmaidle data set has much lower statistical thread latencies than the audio data set, with the worst case for dmaidle being that less than 0.01% of its threads have a latency of 1-1.999 milliseconds. On the other hand, the audio log file has about that many samples with a latency of between 32.000 and 63.999 milliseconds. To find out more about a particular data point, hold the pointer over the data point, as demonstrated in Figure 6.

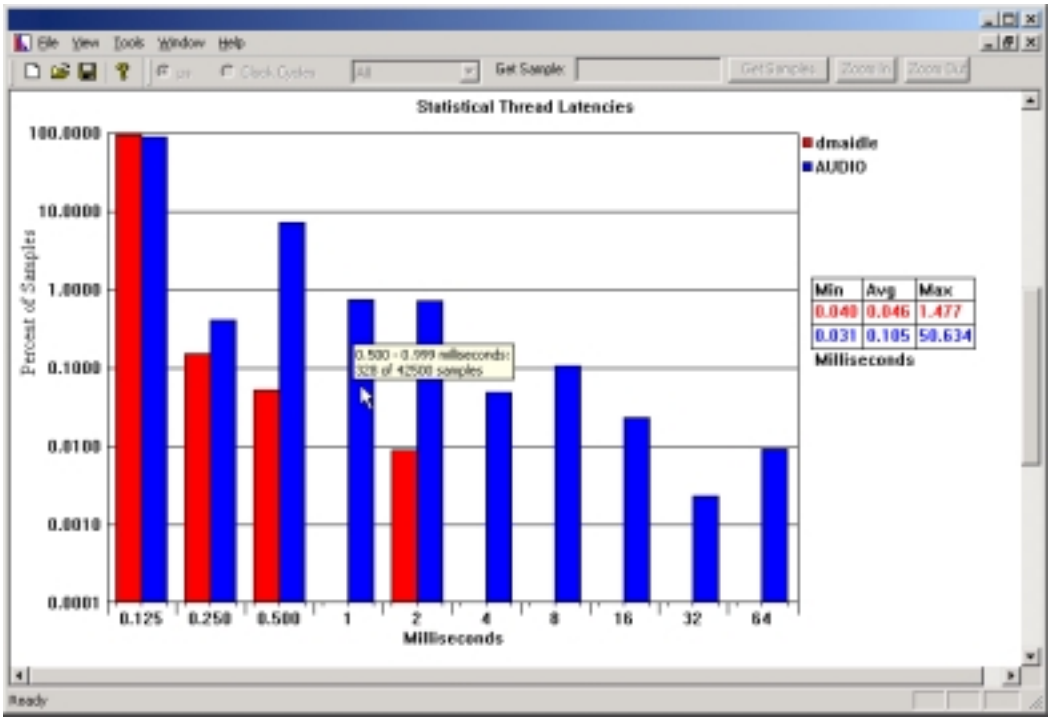

Figure 6. An Example of Comparing Two Data Files

3.3 Causal Analysis

3.3.1 Recording Causal Data

If an unusually large latency appears in the statistical analysis, the Intel® Real-Time Performance Analyzer can help uncover the reason for it. The most straightforward latency cause analysis technique is to observe what the system was doing when the latencies occurred. Whatever was executing most frequently is most likely the culprit. The performance analyzer has the ability to determine exactly what was executing.

During statistical latency sampling, the performance analyzer generates high priority interrupts that store the instruction pointer (EIP) and register state of the interrupted task. When latency in a sample exceeds a user-specified threshold, these data are logged along with the latency sample to allow correlation. To enable causal data collection, create a new document and select Measure Statistical Latencies. If interrupt delays are not enabled, this enables the Log Causal Data box. Select this. You can select the causal data sampling interval. The interval is specified in microseconds, and the default is 500 µs. If this interval is too large, the data may not be very conclusive in identifying the cause of the latency. However, if the interval is too small, the frequent causal data collection can contribute to increasing the latency being measured. For each type of latency, select the threshold of interest. Data is logged if any threshold is exceeded.

Click Start Recording to start data collection. When you've finished recording, view the causal data from a list view by going to View→View Causal Data. This view lists, in

chronological order, each sample that exceeded threshold and the instruction pointers on the stack during that interval. Once this causal data is obtained, use a kernel debugger with symbolic support to associate the most frequent addresses with the corresponding symbols.

3.4 Task Tool

3.4.1 Introduction

The latency characteristics of a system are highly sensitive to the amount of operating system overhead incurred to service any other application concurrently executing on the system. On personal computer and workstation systems the execution environment is highly dynamic and may include a variety of concurrently executing applications.

To test the robustness of a soft device driver or real-time computations with most possible workloads is necessary but can be difficult to implement. Part of the Intel® Real-Time Performance Analyzer is the Task Tool to emulate concurrent activity that could occur on the system in conjunction with the soft device. This allows the developer to simulate various PC platform conditions that could realistically occur and assess their impact on real-time performance. This utility simulates various workloads on the system by consuming processor resources at interrupt, DPC, and various thread priority levels. In this way, the resource consumer can be made to emulate many different kinds of software and device driver loads.

3.4.2 Setting up the Task Tool

Both the main performance analyzer program and the task tool require using the Local Advanced Programmable Interrupt Controller (the Local APIC). By default, it is used by the main program. Control of the local APIC is determined by the following registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SMTKDrv\USELOCALAPIC
```

This is a Boolean value that controls whether or not the main program will use the local APIC. By default, it is set to 1, giving control to the main program. If you set it to 0, then the main program will not use the local APIC, and the Task tool will use it. Because 1 is the default, you can't run the Task Tool until you follow these steps:

1. Set UseLocalAPIC to 0.
2. Restart the system.
3. In a command window do: net start tasktool
4. Click on: Start→Programs→Intel Real-Time Performance Analyzer→TaskTool

You will need to repeat step three every time that the system is rebooted. When the UseLocalAPIC registry value is 0 and the main program is started, the following tools are disabled:

1. Interrupts Masked

2. Log Causal Data

3.4.3 Using the Task Tool

A single common user interface is used to create any arbitrary workload, consisting of periodic ISRs, DPCs or threads. (Periodicity is specified as x microseconds worth of execution time every T milliseconds.) The first dialog (shown in Figure 7) is the main program window. It shows the speed of the processor that the program is running on. Every time you click New, a new task window appears. The task dialog box (see Figure 7) enables you to specify the period and execution time of the task, and (in the case of a thread) the priority. The start button starts the task, and clicking the beep button causes a sound to be played whenever a deadline is missed. You can reset the counter by stopping and restarting the task.

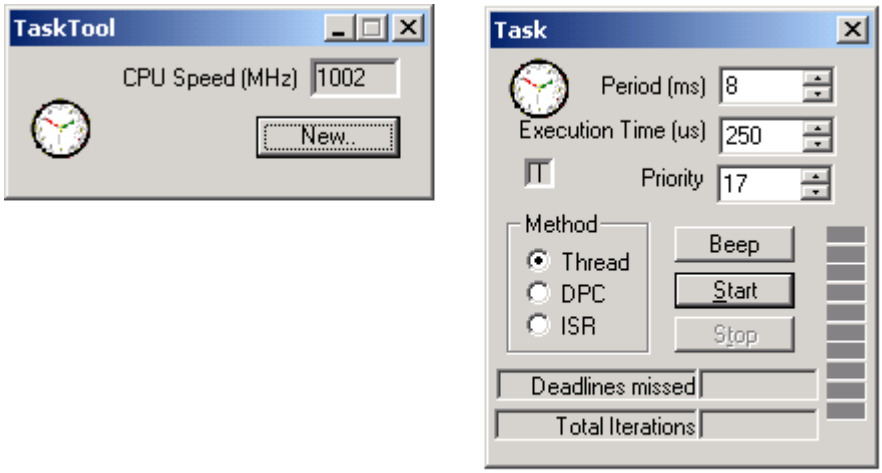

Figure 7. Task Tool and Task Dialog Boxes

4. Program Options, Settings, and Files

4.1 Other Recording Options

Along with choosing what to record, you can choose how to record it. For finer control, click the Advanced Options button to bring up the advanced options dialog (as shown in Figure 8). You can control how often statistical and causal data are collected and how the system data is collected and stored.

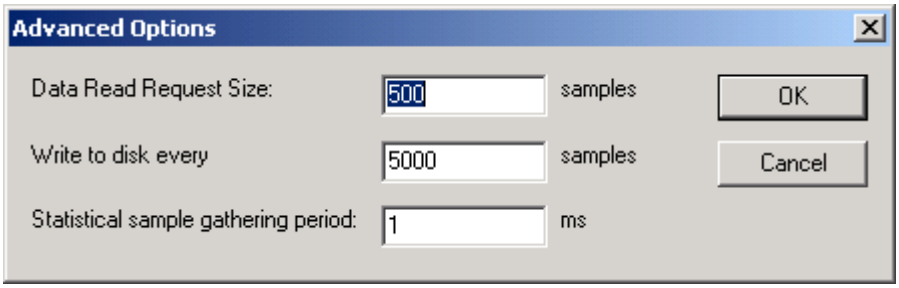

Figure 8. The Advanced Options Dialog

The first option, Data Read Request Size, controls how much data is read from the driver at one time. The default is to read 500 samples at a time; it is not recommended that this be set to a value of more than 5000. The second option controls how often the memory buffer is swapped out to the temporary file on the disk. This must be an integer multiple of the Read Request size. The GUI will enforce this restriction. Note that each sample is 96 bytes. Plan your temporary file locations accordingly. The final option is the statistical timer delay, which controls how often statistical data is collected.

4.2 Program Options

Figure 9 shows the program options dialog. This dialog controls the placement of temporary files, the behavior of the various views, and the program's interaction with other programs in the system.

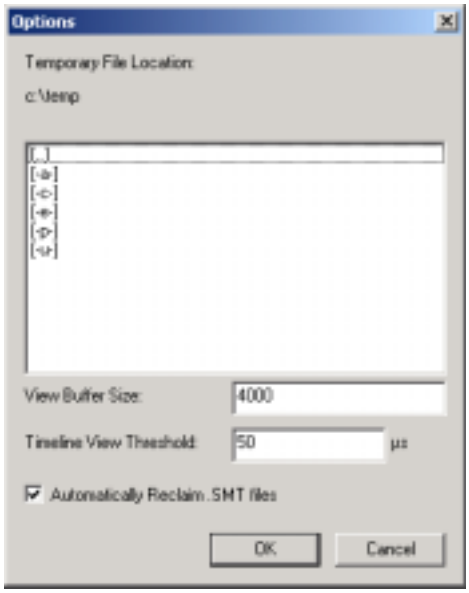

Figure 9. The Program Options Dialog

The large box is for setting the temporary file location. By double clicking on the various directory names, you can select a location where you want to store the temporary files. The view buffer size controls the maximum number of elements to be loaded into the list view at any one time. The timeline view threshold controls the point at which the timeline switches from bar graphs to an actual event-by-event timeline. In this example, whenever the timeline is zoomed in enough that each pixel is less than 50 µs wide, the timeline switches to the event-by-event view.

The last option controls whether .SMT files (the file format used by the Intel® Real-Time Performance Analyzer) are automatically reclaimed. If this box is checked, the registry will be automatically updated to cause .SMT files to be opened by the performance analyzer. You will be able to open .SMT files by double clicking on them in Windows Explorer*. Other applications that use .SMT files might not function correctly. If this box is unchecked, other applications will be able to take control of .SMT files, and you will no longer be able to open them by double-clicking them. If this box is unchecked, then associations with .SMT files will not be removed if the program is uninstalled.

4.3 Cleaning Temporary Files

The Intel® Real-Time Performance Analyzer can create very large temporary files while running. If the program or your system crashes, these files can be left behind and not deleted. They are named SMT???.tmp, where ??? is a hexadecimal number. These files are stored in the directory specified in the Program Options dialog box. You can remove them manually, or via the Tools→Clean Temp Directory menu option, which deletes the files for you. To ensure that files that are in use are not deleted, this option is only available when there is no file open in the performance analyzer program.

4.4 Output Files

There are three primary output file formats. The first is the .SMT file that is the document type saved by the Intel® Real-Time Performance Analyzer. This file contains all of the samples that were gathered, plus information about what was gathered. Intel does not intend for it to be edited or viewed with any application except the performance analyzer. The performance analyzer also can export the contents of several of its views to text files. These are simply the text drawn in the view formatted as plain text. Finally, it can export the various charts in the charted view to bitmap files. When asked to save the files, you are asked for the filename. Each chart is saved as a separate bitmap. The files are saved with that filename plus a suffix that indicates which chart it is. Table 1 lists the chart name to suffix mapping.

Table 1. Chart Name to Suffix Mapping

Chart Name	Suffix
System DPC Execution Time	_SysDPCExec.bmp
System DPC Latency Time	_SysDPCLatency.bmp
System Interrupt Masked Time	_SysInterMaskedTime.bmp
System HIGH IRQL time	_SysHighIrqlTime.bmp
System Work Item Latency Time	_SysWorkItemLatency.bmp
System Work Item Execution Time	_SysWorkItemExecTime.bmp
System ISR Execution Time	_SysISRTime.bmp
Statistical DPC Latency Time	_StatDPCLatency.bmp
Statistical Thread Latency Time	_StatThreadLatency.bmp
Statistical Timer DPC Time	_StatTimerDPCLatency.bmp
Statistical Work Item Time	_StatWorkItemLatency.bmp

4.5 Program Files

Table 2 lists all of the files installed and used by the Intel® Real-Time Performance Analyzer.

Table 2. Program Files

Filename	Description
<Install Directory>\smtktrc.exe	Win32 executable file that implements the main GUI
<Install Directory>\User's Guide.doc	This file
<Install Directory>\Tasktool.exe	Win32 executable that implements the task tool
<Install Directory>\rtexports.h	C Header file that allows your code to insert markers into the data stream
<Install Directory>\smtkdrv.lib	C Library that you must link your driver to when using rtexports.h
<Windows Directory>\System32\Drivers\smtkdrv.sys	WDM driver that measures kernel data
<Windows Directory>\System32\Drivers\tasktool.sys	WDM driver that creates measured tasks
<Windows Directory>\System32\Drivers\smtkstub.dll	Stub functions that allow the performance analyzer to execute on all platforms

5. Inserting Software Markers

The Intel® Real-Time Performance Analyzer can display software-created markers. If you are a driver writer, you might want to be able to see exactly when a certain internal event happened. If so, use software markers.

The performance analyzer's driver (smtkdrv.sys) exports one callable function, RtuneInsertMarker(). This function takes as its parameters five ULONGLONGs, which are unsigned 64-bit integers. The first one is called ID, and the others are called UserVar1 through UserVar4. The performance analyzer places no meaning on these variables. They are displayed as-is in the various views.

To place a marker in the data stream, include RTExports.h (found in the main program directory) in your source code, call RtuneInsertMarker() from your code, and link to SMTKDrv.lib in your driver link step.

These markers appear in both the list and timeline view. When the timeline is zoomed out, a small red flag will appear above any column that has a marker in it. When the timeline is zoomed in, a vertical red line will appear on the graph at the marker point.

6. Error Messages

Refer to this section if you have an error message while running the Intel® Real-Time Performance Analyzer. In all of these cases, an *x* indicates a Windows error code.

6.1 Errors During Program Startup

Error Message	Description
This program requires at least 800x600 display resolution; 1024x768 is strongly recommended.	This message is displayed if trying to run the program on a computer at 640 x 480 or 800 x 600. Change the screen resolution.
This program is already running.	This message is displayed if trying to run two instances of the program simultaneously. Switch to the other instance, or if no other instance exists, reboot your computer.
This program can only run on an Intel Pentium Pro or later.	The Intel® Real-Time Performance Analyzer requires that you use a genuine Intel processor. It must be one of the supported processors mentioned in Section 1.1.

6.2 Errors When Preparing to Record

Error Message	Description
Error *x* opening driver – Is it installed?	This message is displayed if there is an error opening the performance analyzer's driver. The most likely cause is the driver not being installed. Run the installation program again to install it.
There is nothing to record. Please select some options.	This message is displayed if trying to start recording with no recording options selected. Select something to record and try again.
Error *x* getting capabilities Error *x* -- Unable to get processor speed	These messages indicate that the driver did not respond to a request for information correctly, probably indicating a corrupt installation. Reinstall the program.
CreateEvent Failed	This indicates that an attempt to communicate with the driver failed. Reboot the system and try again.
Error Opening Temp File – *x*	This message indicates an error opening the temporary file. Check the location of the temporary file (in Tools→Program options) and try again.

6.3 Errors While Recording

Error Message	Description
Samples Dropped near sample #y	This message indicates that the GUI was unable to read samples fast enough, resulting in dropped samples near the sample with serial number y. This can sometimes be fixed by increasing the read request size in the advanced options dialog box, or by recording less data.
2 gigabyte limit reached – Not recording samples	This message appears when the data file has reached than two gigabytes in size. At this point, data is no longer being recorded, although the min/average/max displays are still correct.
Disk error – Not recording samples	This message indicates an error writing to the disk and usually indicates a hardware problem.

6.4 Errors While Viewing Data

Error Message	Description
Error *x* selecting bitmap	There was an error creating a bitmap to save during the exporting of bitmaps from the chart view.
Sorry, there is a maximum of 5 documents charted at once.	This message displayed if trying to import more than five documents into a charted view.
This document does not have any causal samples.	This message is displayed if trying to view causal data on a program that doesn't have any.

6.5 Errors While Loading a Document

Error Message	Description
Unknown file format. This file can not be loaded by this version of this product.	This error most likely indicates a corrupt .SMT file. The file is probably lost.

7. Tools Availability

Some features are not available depending on a combination of the operating system and processor type. Table 3 summarizes availability for different operating systems:

Table 3. Operating System Based Feature Availability

Feature	Supported by Windows2000 and Windows NT SP6 or later?	Supported by Windows 98?	Supported by Windows Millenium?
System Tools (excluding interrupt holdoff measurement)	Yes	No	No
Interrupt Holdoff measurement	Yes	No	No
Statistical Tools	Yes	Yes	Yes
Causal Data	Yes	Yes	No if USB audio, else Yes
Task Tool	Yes	No	No

Table 4 summarizes feature availability for various Intel processors.

Table 4. Processor Based Feature Availability

Feature	Supported by Celeron, Pentium II and Pentium III processors?	Supported by Pentium 4 processor?	Supported by other processors?
System Tools (excluding Interrupt holdoff measurement)	Yes	Yes	No
Interrupt holdoff measurement	Yes	No	No
Statistical Tools	Yes	Yes	No
Causal Data	Yes	Yes	No
Task Tool	Yes	Yes	No